DIKTAT MATERI

ISB-105 ALGORIMA DAN PEMROGRAMAN

Sofia Umaroh, dkk.

ISB-104

KATA PENGANTAR

Syukur Alhamdulilah penulis ucapkan karena berkat rahmat dan hidayahnya modul Mata Kuliah **"ALGORITMA DAN PEMROGRAMAN"** dapat diselesaikan tepat pada waktunya. Modul ini digunakan untuk praktikum Pemrograman Dasar Semester Ganjil Tahun Ajaran 2023/2024 dan digunakan oleh mahasiswa yang mengambil mata kuliah tersebut. Materi Algoritma dan Pemrograman berisi tentang konsep dasar algoritma dan penerapannya pada pemrograman menggunakan bahasa pemrograman tingkat tinggi dengan menggunakan Bahasa Pascal dan C. Melalui mata kuliah ini diharapkan mahasiswa mampu menyusun algoritma untuk menyelesaikan masalah saintifik dengan membuat permrograman procedural berupa program CLI.

Bandung, 1 Agustus 2023

Ketua Penyusun Mata kuliah Algoritma dan Pemrograman

Sofia Umaroh,S.Pd.,M.T.

Tim Penyusun Algortima dan Pemrograman

Nur Fitrianti Fahrudin, S.Kom., M.T

Kurnia Ramadhan Putra, S.Kom., M.T

Asep Rizal Nurjaman, S.Kom., M.Kom

Mira Musrini Barmawi, S.Si., M.T.

DAFTAR ISI

1. DASAR-DASAR PEMROGRAMAN BAHASA C

CPMK 1	:	*Mampu menunjukkan kinerja mandiri dan bermutu dalam menerapkan konsep dan teori dasar algoritma dan pemrograman berbasis prosedural untuk memecahkan masalah komputasi*
SUB-CPMK 1	:	*Menjelaskan konsep dasar dan representasi algoritma dalam menyelesaikan suatu masalah sederhana tentang saintifik*
SUB-CPMK 2	:	*Menggunakan data (numerik, boolean, teks), variablel, pernyataan dan operasi (aritmatika, teks dan boolean)*

DASAR TEORI

DASAR PEMROGRAMAN BAHASA C

1. BARIS KOMENTAR

Baris komentar adalah baris-baris yang menjelaskan maksud dari perubah yang digunakan atau maksud dari program itu sendiri. Hal ini dimaksudkan untuk memudahkan pelacakan atas perubah yang digunakan apabila program yang digunakan cukup besar atau memudahkan orang lain memahami program yang kita buat. Dalam program, baris komentar diletakkan diantara tanda /* dan */ dan baris ini tidak dikerjakan oleh komputer, hanya dianggap sebagai baris kosong.

2. BENTUK / STRUKTUR PROGRAM C

Bentuk program C mirip dengan kebanyakan program bahasa tingkat tinggi lainnya. Bentuk programnya adalah:

1. Judul Program
2. Daftar Header File
3. Deklarasi
4. Deskripsi

2.1. Judul Program

Judul program sifatnya sebagai dokumentasi saja, tidak signifikan terhadap proses program. Ditulis dalam bentuk baris komentar.

Contoh:

```
/*
Program: Menghitung Luas Segitiga
Author: Sofia
Date: 19/02/2019
*/
```

2.2. Header File

C menyediakan sejumlah file judul (header file) yaitu file yang umumnya berisi prototipe fungsi, definisi makro, variabel dan definisi tipe. File ini mempunyai ciri yaitu namanya diakhiri dengan extension.h.

Contoh:

```
#include <stdio.h>      //menambahkan library std input-output
#define SIZE 400        //mendefinisikan konstanta
```

Keterangan: menyatakan bahwa agar membaca file bernama stdio.h saat pelaksanaan kompilasi.

2.3. Deklarasi

Deklarasi adalah bagian untuk mendefinisikan semua nama yang dipakai dalam program. Nama tersebut dapat berupa nama tetapan (konstanta), nama variabel, nama tipe, nama prosedur, nama fungsi.

Contoh:

```
int alas;                    //tipe bilangan bulat
int tinggi;             //tipe bilangan bulat
float luasSegitiga;          //tipe bilangan decimal
```

2.4. Deskripsi

Bagian inti dari suatu program yang berisi uraian langkah-langkah penyelesaian masalah. Program C pada hakekatnya tersusun atas sejumlah blok fungsi. Sebuah program minimal mengandung sebuah fungsi. Setiap fungsi terdiri dari satu atau beberapa pernyataan, yang secara keseluruhan dimaksudkan untuk melaksanakan tugas khusus. Bagian pernyataan fungsi (disebut tubuh fungsi) diawali dengan tanda { dan diakhiri dengan tanda }.

Deskripsi pada program menghitung luasSegitiga:

```
scanf("%d",&alas);
scanf("%d",&tinggi);
luasSegitiga = alas*tinggi/2;
```

3. VARIABEL

Variabel dalam program digunakan untuk menyimpan suatu nilai tertentu dimana nilai tersebut dapat berubah-ubah. Setiap variabel mempunyai tipe dan hanya data yang bertipe sama dengan tipe variabel yang dapat disimpan di dalam variabel tersebut. Setiap variabel mempunyai nama. Pemisahan antar variabel dilakukan dengan memberikan tanda koma. **Nama variable harus sesuai dengan kegunaan dari variable tersebut.**

Contoh:

```
int alas, tinggi;       //tipe bilangan bulat
float luasSegitiga;     //tipe bilangan desimal
```

Dari contoh diatas, variabel `alas` dan `tinggi` hanya boleh menerima data yang bertipe integer (bulat), tidak boleh menerima data bertipe lainnya. Variabel `luasSegitiga` hanya bisa diisi dengan bilangan float (pecahan).

4. KONSTANTA

Berbeda dengan variabel yang isinya bisa berubah selama eksekusi program berlangsung, nilai suatu konstanta tidak bisa berubah.

Contoh :

```
const int m = 8;
#define pajak 0.05
```

5. OPERATOR

Operator adalah pengendali operasi yang akan dilakukan pada beberapa operan sehingga membentuk sebuah ekspresi. Terdapat 3 macam operator yang bisa digunakan dalam pemrograman, yaitu:

a. Operator aritmatik
   Operator ini untuk membentuk perhitungan aritmatik. Operator yang diapit oleh 2 operan dapat berupa bilangan integer atau real.

Lambang	**Deskrpsi**	**Tipe data operan**	**Algoritma**	**Bahasa C**
*	Perkalian	integer, real	a <- b * c	a = b * c;
/	Pembagian	Integer, real	a <- b / c	a = b / c;
%	Operator sisa pembagian (modulo)	Integer	a <- b mod c	a = b % c;
+	Penjumlahan	Integer, real	a <- a + b	a = b + c;
-	Pengurangan	Integer, real	a <- a – b	a = b - c;

b. Operator assignment
   Operator ini digunakan untuk memasukan nilai ke dalam sebuah variable, tanpa menghilangkan atau mengosongkan nilai variable sebelumnya. Berikut contoh penggunaan operator assignment.

Lambang	**Deskrpsi**	**Tipe data operan**	**Algoritma**	**Bahasa C**
+=	Menambahkan	Integer	x <- x+1	x += 1;
-=	Mengurangkan	Integer	x <- x – 1	x -= 1;
*=	Mengalikan	Integer	x <- x * 1	x *= 1;
/=	Membagi	Integer	x <- x / 1	x /= 1;
%=	Mem-modulasi	Integer	x <- x % 1	x %= 1;

c. Operator relasional
   Operator ini digunakan untuk membandingkan dua operan dan hasilnya berupa nilai Boolean (Benar atau Salah).

Lambang	**Deskrpsi**	**Tipe data operan**	**Algoritma**	**Bahasa C**
==	Sama dengan	Integer, real, char	x == y	x == y;
!=	Tidak sama dengan	Integer, real, char	x <> y	x != y;
>	Lebih dari	Integer, real	x > y	x > y;
<	Kurang dari	Integer, real	x < y	x < y;
>=	Lebih dari sama dengan	Integer, real	x >= y	x >= y;

<=	Kurang dari sama dengan	Integer, real	x <= y	x <= y;

d. Operator logika
Operator ini digunakan untuk mengkombinasikan hasil ekspresi yang mengandung operator rasional.

Lambang	**Deskrpsi**	**Algoritma**	**Bahasa C**
&&	Dan / AND	x > 0 and y > 0	x > 0 && y > 0
||	Atau / OR	x > 0 or y > 0	x > 0 || y > 0
!	Tidak / NOT	!(x > 0)	!(x > 0)

6. FUNGSI main()

Fungsi main() harus ada pada program, karena fungsi inilah yang menjadi titik awal dan titik akhir eksekusi program. Tanda { di awal fungsi menyatakan awal tubuh fungsi sekaligus awal eksekusi program, sedangkan tanda } di akhir fungsi merupakan akhir tubuh fungsi dan sekaligus akhir eksekusi program.

7. FUNGSI printf()

Merupakan fungsi yang digunakan untuk menampilkan data ke layar. Dengan menggunakan fungsi ini, tampilan dapat diatur (diformat) dengan mudah. Bentuk umum dari fungsi ini :

```
printf("string kontrol", argumen1, argumen2, ....);
```

String kontrol dapat berupa keterangan beserta penentu format (seperti %d, %f). Argumen adalah data yang akan ditampilkan, dapat berupa variabel, konstanta, maupun ungkapan.

Contoh :

```
/* Program Satu */
#include <stdio.h>
main()
{
      Printf("Belajar Pemrograman Komputer");
}
```

8. FUNGSI scanf()

Merupakan fungsi yang digunakan untuk menerima data dari yang diinputkan/ dimasukkan dari keyboard.

9. FUNGSI getch()

Merupakan fungsi yang digunakan untuk membaca sebuah karakter yang dimasukkan dari keyboard dan karakter tersebut tidak akan ditampilkan pada layar. Untuk bisa menggunakan fungsi ini file `conio.h` harus disertakan.

10. FUNGSI clrscr()

Merupakan fungsi yang digunakan untuk membersihkan (menghapus) layar. Untuk bisa menggunakan fungsi ini file `conio.h` harus disertakan.

LATIHAN

1. **Latihan 1: Input/Output**

   Menerima Panjang dan lebar, kemudian menghitung luas dan kelilingnya.

```
1.  /*
2.  Program: menghitung luas dan keliling persegi panjang
3.  Author:
4.  Date:
5.  */
6.
7.  #include<stdio.h>
8.  #include<conio.h>
9.
10. int main(){
11. int p,l,luas,keliling;
12.
13. printf("Panjang= ");
14. scanf("%d",&p);
15. printf("Lebar = ");
16. scanf("%d",&l);
17.
18. //proses menghitung luas dan keliling
19. luas = p*l;
20. keliling = 2*(p+l);
21.
22. printf("Luas persegi panjang = %d\n",luas);
23. printf("Keliling persegi Panjang = %d\n",keliling);
24.
25. return 0;
26. }
```

2. **Latihan 2: Operator Aritmatika**

   Menerima total detik dan mengkonversikan total detik menjadi jam menit detik.

```
1.    /*
2.    Program: Konversi total detik ke jam-menit-detik
3.    Author:
4.    Date:
5.    */
6.
7.    #include<stdio.h>
8.
9.    int main(){
10.
11.   int totalDetik;
12.
13.   //Proses baca total detik
14.   printf(”Masukan total detik= ”);
15.   scanf(”%d”,&totalDetik);
16.
17.   //proses konversi
18.   jam = totalDetik/3600;
19.   sisa = totalDetik % 3600;
20.   menit = sisa/60;
21.   detik = sisa % 60;
22.
23.   //cetak hasil konversi
24.   printf("%d Jam: %d Menit: %d Detik\n",jam,menit,detik);
25.
26.   return 0;
27.   }
```

3. Latihan 3: Kondisi

Menerima ordinat dan absis sebuah kemudian menentukan posisi kuadran titi tersebut dan menghitung jarak dari titik pusat O(0,0).

```
1.    /*
2.    Program: Mengecek posisi kuadran suatu titik
3.    Author:
4.    Date:
5.    */
6.
7.    #include<stdio.h>
8.    #include<conio.h>
9.    int main(){
10.
11.   float x,y;
12.   float jarak;
13.
14.   //proses baca ordinat & absis
15.   printf(”Ordinat= ”);
16.   scanf(”%f”,&x);
17.   printf(”Absis = ”);
18.   scanf(”%f”,&y);

19.   //proses mengecek posisi kuadran
20.   if (x>0 && y>0){
21.        printf(“Kuadran I”);
22.   } else if (x<0 && y>0){
23.        printf(“Kuadran II”);
24.   } else if (x<0 && y<0){
25.        printf(“Kuadran III”);
26.   } else if (x>0 && y>0){
27.        printf(“Kuadran IV”);
28.   }
29.
30.   //proses menghitung jarak
31.   jarak = sqrt(pow(P.X-0,2)+pow(P.Y-0,2));
32.
33.   //cetak jarak ke layar
34.   printf(“Jarak titik ke tiitk pusat (0,0) = %.2f”,jarak);
35.
36.   return 0;
37.   }
```

4. Latihan 5: Pengulangan for

Mencetak angka urut menggunakan pengulangan for.

```
1.    /*
2.    Program: Mencetak angka urut
3.    Author:
4.    Date:
5.    */
6.
7.    #include<stdio.h>
8.    int main(){
9.
10.   int i,batasLoop;
11.
12.   //proses baca batasLoop
13.   printf(”Batas Loop = ”);
14.   scanf(”%d”,&batasLoop);
15.
16.   //proses pengurutan angka dari 0 hingga batas loop
17.   for (i=0;i<batasLoop;i++){
18.        printf(“%d”,i);
19.   }
20.
21.   return 0;
22.   }
```

5. Latihan 4: Pengulangan While

Menerima suatu bilangan yang merupakan total jumlah anak ayam, kemudian mencetak lagu anak ayam.

```
1.    /*
2.    Program: Lagu Anak Ayam
3.    Author:
4.    Date:
5.    */
6.
7.    #include <stdio.h>
8.    int main(){
9.    int banyakAyam;
10.
11.   printf("Banyak anak ayam: ");
12.   scanf("%d",&banyakAyam);
13.
14.   while (banyakAyam>1){
15.       printf("Anak ayam turunlah %d\n",banyakAyam);
16.       banyakAyam--;
17.       printf("mati satu tinggalah %d\n",banyakAyam);
18.   }
19.
20.   printf("Anak ayam turunlah %d\n",banyakAyam);
21.   printf("mati satu tinggal induknya\n");
22.
23.   return 0;
24.   }
```

TUGAS

1. Buatlah program yang mencetak kalimat "Saya yakin pasti bisa belajar Pemorgraman Dasar!" sebanyak 100 kali.

2. Buatlah program dalam bahasa C yang menerima satu angka yang terdiri dari lima digit dari pengguna, kemudian pisahkan angka ke digit masing-masing dan cetak digit yang dipisahkan satu sama lain dengan masing-masing lima spasi. Misalnya, jika pengguna mengetikkan nomor 42339, program harus mencetak:

```
9     3     3     2     4
```

3. Buatlah program yang menerima indeks nilai suatu mata kuliah, kemudian mencetak predikat nilai tersebut dengan ketentuan:

Nilai	Predikat
A	Amat baik
AB	Antara Baik dan Amat baik
B	Baik
BC	Antara cukup dan baik
C	Cukup
D	Hampir cukup
E	Gagal

2. STRUKTUR

SUB-CPMK 3 : *Menggunakan alur logika yang berupa: a) struktur sederhana, b) struktur bersyarat, c) struktur berulang*

DASAR TEORI

RUNTUNAN (SEQUENCE)

Secara umum kode baris program dibaca dan dieksekusi secara berurutan baris demi bari. Misalnya pada algoritma berikut ini:

```
2.  /*
3.  Program: Runtutan
4.  Author:
5.  Date:
6.  */
7.
8.  //Header file
9.  #include<stdio.h>
10.
11. int main() {
12.
13. //deklarasi
14. int a,b,c,d ;
15.
16. //deskripsi
17. a = 3;
18. b = 2;
19. c = a * b;
20. a = 5;
21. d = a + b;
22. printf("%d %d",c,d);
23.
24. return 0;
25. }
```

Analisis:

Perhatikan kode program pada baris ke 18. Variable c diisi dengan hasil perkalian variable a dan b, dimana a = 3, dan b = 2. Sehingga nilai variable c adalah 6.

Pada baris ke-19, variable a diisi dengan 5, maka nilai variable a yang awalnya 3, berubah menjadi 5.

Pada baris ke-20, variable d diisi dengan hasil penjumlahan variable a dan b, sehingga nilai variable d = 5 + 2 = 7

Pada baris ke-21, cetak variable c dan d, dimana c = 6, dan d = 7, maka output dari program di atas adalah:

```
6,7
```

PEMILIHAN (SELECTION)

1. DEFINISI PEMILIHAN IF

Seringkali kita dihadapkan pada beberapa pilihan yang menuntut kita untuk mempertimbangankan kondisi tertentu. Apabila saya lapar, maka saya makan. Pada contoh tersebut, 'saya lapar' adalah kondisi yang menjadi pertimbangan untuk melakukan aksi 'saya makan'. Jika kondisi 'saya lapar' terpenuhi (bernilai benar), maka aksi 'saya makan' dilakukan.

Bentuk pemilihan:

```
if (saya lapar){
    saya makan
}
```

2. STRUKTUR PEMILIHAN

Bentuk pemilihan IF atau dikenal dengan IF statement memiliki bentuk umum sebagai berikut:

```
if (kondisi1){
    (aksi-1)
}else{
    (aksi-2)
} //endif
```

Bentuk pemilihan di atas menunjukkan apabila kondisi1 terpenuhi (bernilai benar), maka aksi-1 dilakukan. Namun apabila kondisi1 tidak terpenuhi (bernilai salah) maka aksi-2 harus dilakukan.

3. PEMILIHAN MAJEMUK

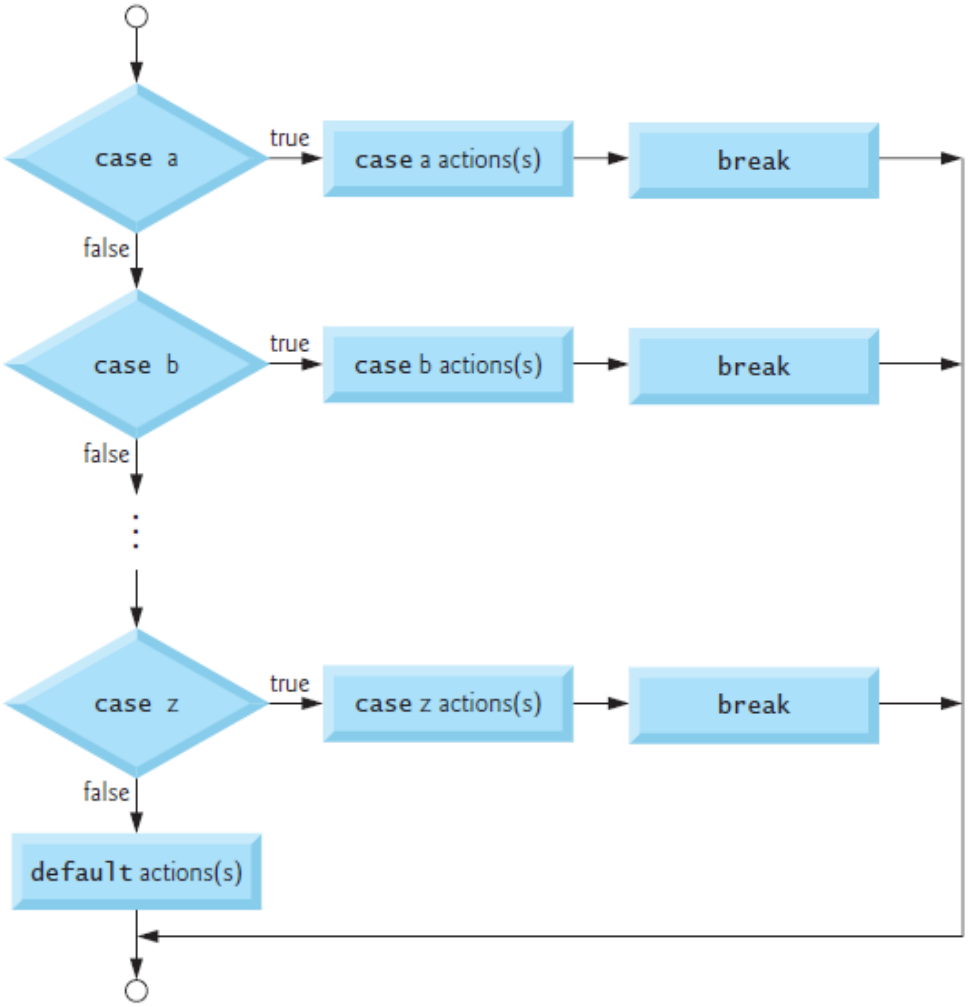
Pemilihan bentuk majemuk (nested-if) digunakan untuk menentukan aksi yang akan dilakukan dengan menyediakan lebihi dari 2 alternatif aksi. Struktur pemilihan mejemuk sebagai berikut:

```
if kondisiA {
     if kondisiB {
         true statementB
     }else{
         false statementB
     }
}else{
     false statementA
}
```

4. Switch…Case Statement

Switch—case merupakan jenis seleksi yang dirancangan khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah atau banyak alternatif penyelesaian. Pernyataan switch—case ini memiliki kegunaan sama seperti if—else bertingkat, tetapi penggunaannya untuk memeriksa data yang bertipe karakter atau integer.

Flowchar Switch..Case



Contoh program:

```
#include <stdio.h>
int main()
{
     char ch='b';
     switch (ch)
     {
         case 'd':
            printf("CaseD ");
            break;
```

```
        case 'b':
           printf("CaseB");
           break;
        case 'c':
           printf("CaseC");
           break;
        case 'z':
           printf("CaseZ ");
           break;
        default:
           printf("Default ");
    }
    return 0;
}
```

Output:

```
CaseB
```

Valid expressions for switch –

```
switch(1+2+23)
switch(1*2+3%4)
```

Invalid switch expressions –

```
switch(ab+cd)
switch(a+b+c)
```

PENGULANGAN (FOR)

Struktur FOR digunakan bila kita mengetahui secara pasti banyaknya pengulangan yang akan dilakukan. Pernyataan FOR mempunyai 3 parameter yaitu :

1. Nilai awal (initial value)
2. Test kondisi yang menentukan akhir loop (condition expression)
5. penentu perubahan nilai (incremental expression)

Bentuk FOR :

```
for (initial value; condition expression; incremental expression)
```

Keterangan:

Initial value: memberikan nilai awal pada variabel kontrol

Condition expression: ekspresi yang menyatakan berhentinya pengulangan. Jika tes kondisi bernilai salah maka loop akan berhenti.
Incremental expression: berfungsi menaikkan/menurunkan nilai dari variabel kontrol. Dapat berupa nilai positif (penaikan) / nilai negatif (penurunan)
Penaikan: setiap loop operator ++ akan menambah nilai 1 ke variabel kontrol
Penurunan: setiap operator -- akan menurunkan nilai 1 pada variabel kontrol

Contoh pengulangan for sebagai berikut:

```
int I;


for (i = 0;i<10;i++){
    printf("%d ",i);
}
```

Kode di atas akan menghasilkan output sebagai berikut:

```
0 1 2 3 4 5 6 7 8 9 10
```

PENGULANGAN BERSARANG

Nested loop atau pengulangan bersarang adalah pengulangan yang ada di dalam pengulangan.

Contoh syntax untuk **nested for loop** :

```
for ( init; condition; increment ) {

   for ( init; condition; increment ) {
      statement(s);
   }
   statement(s);
}
```

Contoh syntax untuk **nested while loop** :

```
while(condition) {

   while(condition) {
```

```
      statement(s);
   }
   statement(s);
}
```

Contoh syntax untuk **nested do...while loop** :

```
do {
   statement(s);

   do {
      statement(s);
   }while( condition );

}while( condition );
```

Catatan tentang loop nesting adalah Anda bisa meletakkan semua jenis loop di dalam tipe loop lainnya. Misalnya, loop 'for' dapat berada di dalam loop 'while' atau sebaliknya.

Contoh nested loop :

```
#include <stdio.h>
int main () {
   /* local variable definition */
   int i, j;
   for(i = 2; i<100; i++) {

      for(j = 2; j <= (i/j); j++)
      if(!(i%j)) break; // if factor found, not prime
      if(j > (i/j)) printf("%d is prime\n", i);
   }
   return 0;
}
```

OUTPUT :

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
```

```
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
53 is prime
59 is prime
61 is prime
67 is prime
71 is prime
73 is prime
79 is prime
83 is prime
89 is prime
97 is prime
```

LATIHAN

Latihan 1 : Switch -case

```
1. #include<stdio.h>
2. void main( )
3. {
4.     int a, b, c, choice;
5.     while(choice != 3)
6.     {
7.         /* Printing the available options */
8.         printf("\n 1. Press 1 for addition");
9.         printf("\n 2. Press 2 for subtraction");
10.          printf("\n Enter your choice");
11.          /* Taking users input */
12.          scanf("%d", &choice);
13.
14.          switch(choice)
15.          {
16.              case 1:
17.                  printf("Enter 2 numbers");
18.                  scanf("%d%d", &a, &b);
```

```
19.                 c = a + b;
20.                 printf("%d", c);
21.                 break;
22.             case 2:
23.                 printf("Enter 2 numbers");
24.                 scanf("%d%d", &a, &b);
25.                 c = a - b;
26.                 printf("%d", c);
27.                 break;
28.             default:
29.                 printf("you have passed a wrong key");
30.                 printf("\n press any key to continue");
31.         }
32.     }
33.}
```

Latihan 2 : Print gender (Male/Female) program according to given M/F using switch

```
#include <stdio.h>

int main()
{
    char gender;

    printf("Enter gender (M/m or F/f): ");
    scanf("%c",&gender);

    switch(gender)
    {
        case 'M':
        case 'm':
            printf("Male.");
            break;
        case 'F':
        case 'f':
            printf("Female.");
            break;
        default:
            printf("Unspecified Gender.");
```

```
    }
    printf("\n");
    return 0;
}
```

Latihan 3 : Nested Loop

```
#include <stdio.h>
int main()
{
    int i, j, rows;

    printf("Enter number of rows: ");
    scanf("%d",&rows);

    for(i=1; i<=rows; ++i)
    {
        for(j=1; j<=i; ++j)
        {
            printf("%d ",j);
        }
        printf("\n");
    }
    return 0;
}
```

TUGAS PRAKTIKUM

1. Gaji yang diterima pegawai terdiri dari gaji pokok, tunjangan dan upah lembur,besar tunjangan 15% dari gaji pokok, besar upah lembur perjam 3% dari gaji pokok.Setiap pegawai memiliki jabatan dan setiap jabatan memiliki bonus yang berbeda :
   - Kepala departemen bonus = Rp.5.000.000
   - Manajer bonus = Rp.3.000.000

- Sales/Marketing bonus = Rp.1.000.000
- SPG/SPB bonus = Rp.500.000
- Buruh bonus = Rp.100.000

Buatlah program menggunakan switch case untuk menghitung gaji yang diterima pegawai . Masukan (Input) : Pangkat ,Gaji,jumlah jam lembur. Keluaran (Output): Gaji pokok,tunjangan,uang lembur,total gaji.

2. Buat program dengan tampilan sebagai berikut:

```
* * * * *
* * * *
* * *
* *
*
```

3. Buat program dengan tampilan sebagai berikut :

```
    *
   * * *
  * * * * *
* * * * * * *
```

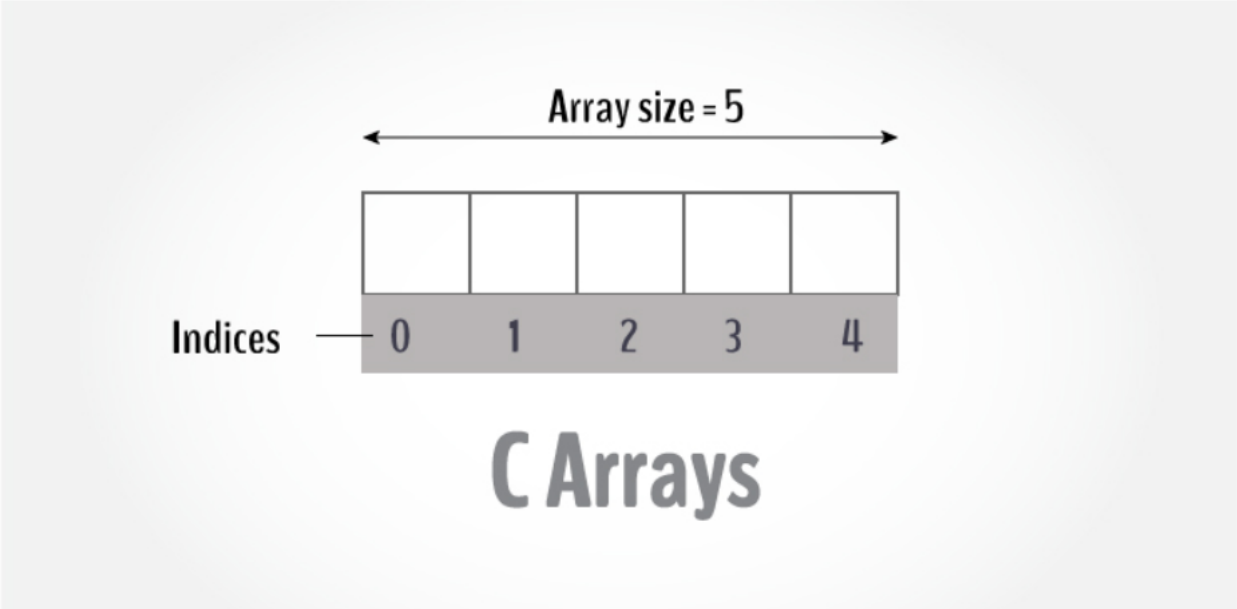
3. ARRAY

SUB-CPMK 4 : *Mampu menggunakan struktur data larik dalam penyusunan algoritma (array).*

DASAR TEORI

ARRAY

Array merupakan koleksi data dimana setiap elemen memakai nama dan tipe yang sama serta setiap elemen diakses dengan membedakan indeks arraynya.
Bentuk : `tipe nama_var[ukuran];`
Keterangan :
Tipe : menyatakan jenis elemen array misal int, char, dll
Ukuran : menyatakan jumlah maksimal elemen array
Contoh : int nilai [5]



Inisialisasi :

```
int mark[5] = {19, 10, 8, 17, 9};
```

ARRAY 2 DIMENSI

```
float x[3][4];
```

Di sini, x adalah array dua dimensi (2d). array x dapat menampung 12 elemen. Bisa dianggap sebagai tabel dengan 3 baris dan setiap baris memiliki 4 kolom.

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

ANIMASI ARRAY

```
/* Program Duapuluh */
/* Program membuat animasi huruf */
#include<stdio.h>
#include<time.h>
#include<conio.h>
#include<windows.h>
void gotoxy(int x, int y) {
COORD pos;
pos.X=x;
pos.Y=y;
SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), pos);
}
void delay(unsigned int mseconds)
{
    clock_t goal = mseconds + clock();
    while (goal > clock());
}

main()
{
```

```
char huruf[7]="Hore..!";
int x,y;
system("cls");
/* ------menampilkan huruf satu persatu kekanan-------*/
for (x=0;x<=6;x++)
{ gotoxy(4+x,4);printf("%c",huruf[x]);
delay(500);
}
/* ------huruf bergerak ke kekanan-------*/
for(x=0;x<=30;x++)
{
gotoxy(4+x,4);printf("Hore..!");
gotoxy(3+x,4);printf(" ");
delay(100);
}
/* ------huruf bergerak ke bawah-------*/
for(x=1;x<=6;x++)
{
for(y=1;y<=20;y++)
{
gotoxy(34+x,3+y);printf(" ");
gotoxy(34+x,4+y);printf("Hore..!");
if (y==20)
gotoxy(34,4+y);printf(" ");
delay(50);
system("cls");
}
}
getche();
}
```

LATIHAN

Latihan 1 : Mencari rata-rata dari jumlah n

```
#include <stdio.h>
int main()
{
     int marks[10], i, n, sum = 0, average;
     printf("Enter n: ");
     scanf("%d", &n);
```

```
     for(i=0; i<n; ++i)
     {
          printf("Enter number%d: ",i+1);
          scanf("%d", &marks[i]);
          sum += marks[i];
     }
     average = sum/n;

     printf("Average = %d", average);

     return 0;
}
```

Latihan 2 : Menyimpan suhu 2 kota selama seminggu dan tampilkan

```
#include <stdio.h>

const int CITY = 2;
const int WEEK = 7;

int main()
{
    int temperature[CITY][WEEK];
    for (int i = 0; i < CITY; ++i) {
        for(int j = 0; j < WEEK; ++j) {
            printf("City %d, Day %d: ", i+1, j+1);
            scanf("%d", &temperature[i][j]);
        }
    }

    printf("\nDisplaying values: \n\n");
    for (int i = 0; i < CITY; ++i) {
        for(int j = 0; j < WEEK; ++j)
        {
```

```
            printf("City %d, Day %d = %d\n", i+1, j+1,
temperature[i][j]);
        }
    }
    return 0;
}
```

Latihan 3 : Nested Loop

```
/* Program membuat matriks */
#include<stdio.h>
main()
{
int A[100] [100];
int m,n,i,j;
/* ------menentukan banyaknya baris & kolom matriks ------*/


printf("Matriks berordo m x n \n");
printf("---------------------- \n\n");
printf("Masukkan banyaknya baris (m) : ");
scanf("%d", &m);
printf("Masukkan banyaknya kolom (n) : ");
scanf("%d", &n);
printf("\n");
/* ------input elemen matriks ------*/
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{printf("Elemen matriks A[%d,%d] : ",i+1,j+1);
scanf("%d",&A[i][j]);
}
}
/* ------menampilkan elemen matriks ------*/
printf("\n");
printf("Matriks A = \n\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("%3d",A[i][j]);
}
printf("\n");
}
getch();
```

```
}
```

TUGAS PRAKTIKUM

1. Buat Program C untuk menemukan jumlah dua matriks orde 2 * 2.
2. Diberikan sebuah masukkan array S sebanyak N elemen String. Anda diminta untuk mencetak semua elemen array S yang memiliki panjang kata genap dan huruf akhirnya adalah "A".

   Input Format

   Baris pertama sebuah bilangan N yang merupakan panjang array S . Baris kedua adalah isi array S sebanyak N

   Output Format

   Tampilkan isi array S dengan ketentuan pada Problem Statement.

   Sample Input

   ```
   4
   Fahri
   Andika
   Intania
   Derina
   ```

   Sample Output

   ```
   Andika
   Derina
   ```

   Explanation

   Elemen array S yang ditampilkan adalah yang memiliki panjang kata genap dan huruf akhirnya "A".

3. Buat animasi gambar dibawah ini bergerak secara diagonal :

   ```
   *
   * *
   * * *
   ```

4. FUNGSI DAN PROSEDUR

SUB-CPMK : *Mampu menyusun program menggunakan subprogram (prosedur dan fungsi) menggunakan bahasa pemrograman tingkat tinggi*

DASAR TEORI

FUNGSI

Cara terbaik untuk menangani program besar adalah menyusunnya dari potongan-potongan program yang berukuran kecil-kecil (disebut modul) merupakan konsep dari pemrograman terstruktur yaitu pemrograman yang menitikberatkan pada pemecahan masalah yang kompleks menjadi masalah yang sederhana.
Program yang terdiri dari modul/subprogram/prosedur/routine lebih mudah ditangani dibanding dengan program yang terdiri dari banyak sekali baris.

- Modul program dalam C disebut fungsi (function)
- Fungsi adalah blok dari kode yang dirancang untuk melakukan tugas khusus.
- Tujuan pembuatan fungsi :
  - program menjadi terstruktur
  - menghemat kode program karena dapat mengurangi duplikasi kode
  - fungsi dapat dipanggil dari program atau fungsi yang lain
  - mempersingkat/memperpendek panjang program
  - mempermudah cek kesalahan

PENDEKLARASIAN DAN PENDEFINISIAN FUNGSI

- Fungsi harus dideklarasikan di dalam program pemanggil/program utama, dengan tujuan supaya program pemanggil mengenal nama fungsi tersebut serta cara mengaksesnya.
- Deklarasi fungsi diakhiri ;
- Fungsi didefinisikan dengan diawali tipe data keluaran fungsi (di depan nama fungsi), defaultnya adalah integer.
- Pendefinisian fungsi tidak diakhiri ;
- Aturan pemberian nama fungsi sama dengan aturan penulisan variabel
- Blok fungsi diawali dengan { dan diakhiri dengan }

Bentuk:

```
tipe_data nama_fungsi (daftar parameter)
```

Keterangan:

daftar parameter : berisi variabel dan tipe variabel yang berfungsi sebagai masukan untuk fungsi

tersebut. Masukan tersebut akan diproses untuk menghasilkan nilai tertentu sesuai dengan tipe data fungsi.

contoh : `int tukar (int x, int y)`

VARIABEL GLOBAL, VARIABEL LOKAL, VARIABEL STATIK

VARIABEL GLOBAL

- Variabel yang dideklarasikan di luar blok fungsi dan bersifat dikenali oleh semua bagian program.
- Data-data yang tersimpan dalam sebuah variabel dapat diakses di setiap blok fungsi
- Disarankan untuk tidak digunakan, karena variabel ini dapat men-sharing-kan data dan dapat diubah secara tidak sengaja oleh suatu blok fungsi, sehingga nilainya bisa berubah.

VARIABEL LOKAL

- Variabel yang dideklarasikan dalam suatu blok fungsi tertentu dan hanya dikenal oleh blok fungsi tersebut.
- Variabel lokal akan dihapus dari memori jika proses sudah meninggalkan blok letak variabel lokalnya.

VARIABEL STATIK

- Variabel statik sering dipakai sebagai variabel lokal.
- Beda variabel lokal dan variabel statik adalah variabel lokal akan dihapus dari memori jika proses sudah meninggalkan blok letak variabel lokalnya, sedangkan variabel statik akan dihapus dari memori jika program dimatikan.

Contoh :

```
int i,j; /* variabel global */
main ()
{
int k,l; /* variabel lokal */
}
fungsi()
{
static int m,n; /* variabel statik */
}
```

PARAMETER FORMAL & PARAMETER AKTUAL (NYATA)

- Parameter formal adalah variabel yang ada pada daftar parameter dalam definisi fungsi.

- Parameter aktual (nyata) adalah parameter yang dapat berupa variabel atau konstanta maupun ungkapan yang dipakai dalam pemanggilan fungsi.

Cara melewatkan/mengirim parameter ke dalam fungsi :

- pengiriman parameter dengan nilai (paramater by value) □□disebut juga parameter masukan

  - Nilai dari parameter aktual akan disalin ke parameter formal.

  - Nilai parameter aktual tidak dapat berubah sekalipun nilai parameter formal berubah-ubah.

  Contoh :

  A,B parameter aktual

  x, y parameter formal

  A x

  B y

  Pada saat pemanggilan suatu fungsi, misal :

  A bernilai 20 □□x juga bernilai 20

  B bernilai 30 □□y juga bernilai 30

- pengiriman parameter secara acuan (parameter by reference) □□disebut juga parameter masukan/keluaran

  - perubahan – perubahan yang terjadi pada nilai parameter formal di fungsi akan mempengaruhi nilai parameter aktual.

  - merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi

  - dipakai untuk mengubah isi suatu varabel di luar fungsi dengan pelaksanaan pengubahan dilakukan di dalam fungsi

FUNGSI YANG MENGEMBALIKAN NILAI

- Pada dasarnya semua fungsi mengembalikan nilai, tetapi untuk fungsi yang tidak mengembalikan nilai disebut fungsi bertipe **void**.

- Untuk mengembalikan nilai sebuah fungsi, digunakan kata **return** yang diikuti dengan nilai yang akan dikembalikan.
- Dalam sebuah fungsi return bisa mengembalikan beberapa nilai, tetapi setiap kata return hanya bisa mengembalikan sebuah nilai saja.

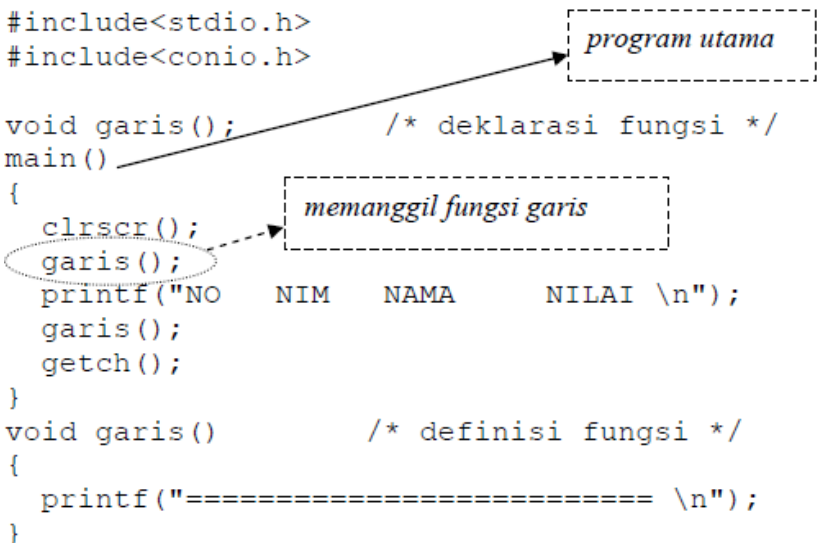
Contoh menggunakan fungsi yang tidak mengembalikan nilai:

```
Algoritma fungsi_garis
Deklarasi
    function garis () → integer
Deskripsi
    garis()
    write("NO  NIM  NAMA  NILAI")
    garis()


function garis() → integer
Deklarasi
Deskripsi
    write("====================")
```

```
Output program :

====================
NO  NIM  NAMA  NILAI
====================
```

LATIHAN

1. Fungsi yang tidak mengembalikan nilai




```
#include<stdio.h>
#include<conio.h>

void garis();          /* deklarasi fungsi */
main()
{
  clrscr();
  garis();
  printf("NO   NIM   NAMA     NILAI \n");
  garis();
  getch();
}
void garis()          /* definisi fungsi */
{
  printf("========================= \n");
}
```

2. Menggunakan fungsi dengan parameter by value

```
/* Program tukar menggunakan fungsi */
/* dengan parameter by value */
#include<stdio.h>
#include<conio.h>
```

```
#include<windows.h>
void tukar(int x,int y); /*deklarasi fungsi*/
main()
{
int a,b;
system("cls");
a=80;
b=11;
printf("Nilai sebelum pemanggilan fungsi \n");
printf("a = %i b = %i \n",a,b);
tukar(a,b);
printf("Nilai setelah pemanggilan fungsi \n");
printf("a = %i b = %i \n",a,b);
getch();
}
void tukar(int px,int py) /* definisi fungsi */
{
int z;
z=px;
px=py;
py=z;
printf("Nilai diakhir fungsi \n");
printf("px = %i py = %i \n",px,py);
}
```

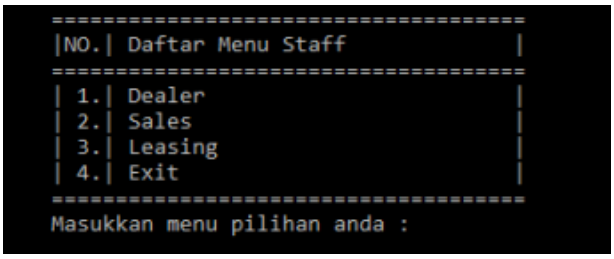
3. Menentukan sebuah nilai apakah ganjil atau genap

```
#include<stdio.h>
main()
{
    int a,b;
    printf("Masukkan nilai = ");scanf("%d",&a); //input a
    b=ganjil(a); //pemanggilan fungsi
    if (b==1)
        printf("Bilangan ganjil\n"); //tampil jika b=1
    else if (b==0)
        printf("Bilangan genap\n"); //tampil jika b=0
    system("pause"); //untuk mempause program
}
```

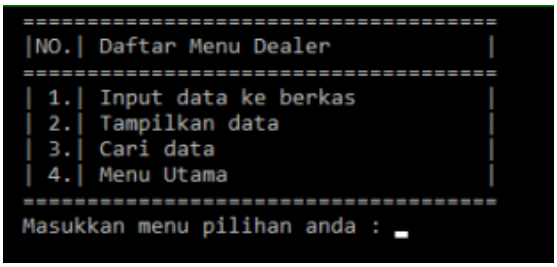
```
int ganjil(int a)
{
    char ganjil,genap;
    if (a%2==1)
        return(1); //mengembalikan nilai 1
    else
        return(0); //mengembalikan nilai 0
}
```

TUGAS

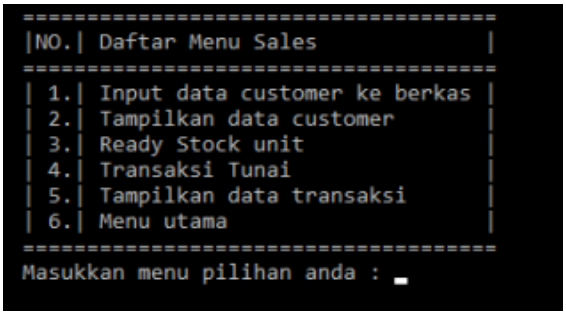
1. Buatlah program berbentuk menu seperti dibawah ini dengan fungsi :
    a. Menu utama



    b. Menu Dealer



    c. Menu Sales



2. Buat sebuah menu program dengan fungsi untuk menghitung :
    a. Balok
    b. Prisma Segitiga

c. Limas Segi Empat
d. Bola

5. ABSTRACT DATA TYPE

SUB-CPMK 4 : *Mampu menggunakan struktur data larik dalam penyusunan algoritma (array).*

DASAR TEORI

ADT

Abstract Data File(ADT) adalah struktur data dan operasinya yang dikemas menjadi entitas. Sebuah program dalam bahasa C yang "utuh", seringkali terdiri dari beberapa modul program. Program yang dibagi-bagi menjadi beberapa file seharusnya dapat dikompilasi terpisah (setiap modul membentuk sebuah object code). Dengan demikian, pembuatan sebuah executable code dapat dilakukan dengan melakukan link terhadap sejumlah object code yang sudah dikompilasi (penghematan waktu dan duplikasi usaha, reusability) Supaya dapat dikompilasi dengan benar, modul yang lain dapat dilakukan dengan melakukan include terhadap file header. File header dalam bahasa C adalah sebuah file yang hanya berisi deklarasi type dan prorotype fungsi ("tidak boleh" ada deklarasi variabel). Standard penulisan file header dapat dilihat pada contoh program kecil.

Contoh program lengkap dengan beberapa modul yang diletakkan dalam beberapa file dapat dilihat pada Contoh Program kecil untuk pemanfaatan ADT JAM yang didefinisikan.

Skema program yang dipecah menjadi beberapa file.

Pada hakekatnya sebuah program utuh terdiri dari kelompok file sebagai berikut:

1. File header, dengan nama xxx.h. Untuk setiap type dan primitifnya, ada sebuah file xxx.h. Contoh: jika anda memerlukan ADT JAM, DATE dan mesin KATA maka ada 3 buah file header yaitu Jam.h, DATE.h dan KATA.h
2. File yang berisi BODY dari File header yang bersangkutan: xxx.c. File ini disebut sebagai file realisasi dari prototype yang didefinisikan pada

xxx.h. Akan ada sebuah xxx.c untuk setiap xxx.h. Untuk contoh di atas, akan ada JAM.c, DATE.c dan KATA.c.

3. File yang berisi main program (dan prosedur/fungsi lain yang hanya dibutuhkan oleh main), misalnya dengan nama main.c Jadi sebuah program utuh akan terdiri dari sebuha main.c,sebuah xxx.h dan xxx.c Skema penulisan program utuh untuk setiap jenis file diberikan sebagai berikut

1. File header :

```
/* File xxx.h */
/* Deskripsi : keterangan isi file header */
/* Isi : deklarasi type dan 37xecutab */
/* deklarasi TYPE dan PROTOTYPE */
/* File header tidak boleh mengandung deklarasi 37xecutab!!! */
/* */
#ifndef xxx_h
#define xxx_h
/* Bagian I : berisi deklarasi konstanta */

/* Bagian II : berisi deklarasi TYPE */

/* Bagian III : berisi deklarasi prototype prosedur dan fungsi*/
/* yang merupakan 37xecutabl TYPE tsb */
/* Kelompokkan fungsi dan prosedur sesuai dengan standard di kelas*/
/* Misalnya : konstruktor, 37xecutab, predikat, operator
relasional*/
/* operator aritmatika, operator lain dsb */
#endif
```

2. File BODY :

```
/* File xxx.c */
/* Deskripsi : keterangan isi file body */
/* Isi : realisasi/kode program dari semua prototype */
/* yang didefinisikan pada xxx.h */
/* Untuk sebuah MESIN, akan mengandung deklarasi state 37xecutab */
```

```
/* dari mesin ybs */

#include “xxx.h”
/* Realisasi kode program, sesuai urutan pada xxx.h */
/* Copy dari xxx.h, kemudian edit */
```

3. File main program

```
/* File : main.c */
/* Deskripsi : program utama dan semua nama 38xecu thd persoalan*/

#include “xxx.h”
/* include file lain yang perlu */

/* Bagian I : berisi kamus GLOBAL dan Prototypr */
/* deklarasi semua nama dan prosedur/fungsi global*/

/* Bagian II : PROGRAM UTAMA */
int main () {
/* Kamus 38xecu terhadap main */

/* Algoritma */
return 0;
}

/* Bagian III : berisi realisasi kode program yang merupakan */
/* BODY dari semua prototype yang didefinisikan pada file ini */
/* yaitu pada bagian I, dengan urut-urutan yang sama */
/* Copy prototype, kemudian edit!! */
```

Untuk memproses, lakukan kompilasi terpisah untuk xxx.c dan main.c, kemudian link untuk membentuk sebuah xecutable file. Pada kuliah ini, untuk setiap xxx.h dan xxx.c, dibuat main program yang sengaja dibuat untuk mentest setiap prosedur dan fungsi yang dibuat, yang disebut sebagai “driver”, atau “test stub”. Xxx.h, xx.c dan mxxx.c disimpan dalam sebuah

direktori dan hasil test juga disimpan dalam direktori yang sama dan terpisah. Dengan cara semacam ini, sebuah modul yang sudah ditest dengan baik dapat digunakan ulang secara efisien, dan terdokumentsi dengan rapi.

TIPE DATA BENTUKAN

Struktur adalah koleksi dari variabel yang dinyatakan dengan sebuah nama, dengan sifat setiap variabel dapat memiliki tipe yang berlainan. Struktur bisa dipakai untuk mengelompokkan beberapa informasi yang berkaitan menjadi sebuah kesatuan (tipe data bentukan).

Contoh sebuah struktur adalah informasi data tanggal, yang berisi:

- Tanggal
- Bulan, dan
- Tahun.

1. Bentuk umum mendeklarasikan/mendefinisikan struktur

Struct nama_tipe_struktur

```
{tipe field1;
  tipe field2;
  .....................
  .....................
  tipe fieldN;
}variabel_struktur1,...,variabel_strukturM;
```

Adapun **variabel_struktur1,...,variabel_strukturM** menyatakan bahwa variabel struktur yang dideklarasikan bisa lebih dari satu.

Contoh mendefinisikan struktur

```
struct data_tunggal
{int tanggal;
```

```
  int bulan;
  int tahun;
};tgl_lahir;
```

Yang mendefinisikan tipe struktur bernama data_tunggal, yang terdiri dari tiga buah elemen (field) berupa tanggal, bulan dan tahun. Field adalah sebutan untuk elemen struktur. Sedangkan variabel tgl_lahir betipe struktur data_tunggal yang mengandung tiga field yaitu tanggal, bulan dan tahun.

Note: nama_tipe_struktur atau variabel_struktur boleh dihilangkan tetapi tidak boleh kedua-duanya dihilangkan.

2. Pemanggilan elemen struktur

Elemen struktur dapat dipanggil dalam program menggunakan bentuk

Variabel_struktur.nama_field

Antara variabel_struktur dan nama_field dipisahkan dengan operator titik (disebut operator titik anggota struktur).

Sedangkan untuk memberikan data nama ke field nama, pernyataan yang diperlukan misalnya berupa

```
strcpy(info_rekan.nama,"Ummu Habibah");
```

Contoh Program :

```
#include"stdio.h"
#include"conio.h"
#include"string.h"
struct data_tanggal
{int tanggal;
 int bulan;
 int tahun;
 };

 struct data_rekan   //definisi tipe data_rekan
```

```
 {char nama[31];
  struct data_tanggal tgl_lahir;
  };
void main()
{ //definisi tipe data_tunggal

  struct data_rekan info_rekan; //deklarasi variabel

  strcpy(info_rekan.nama,"UMMU HABIBAH");

  info_rekan.tgl_lahir.tanggal=15;
  info_rekan.tgl_lahir.bulan=5;
  info_rekan.tgl_lahir.tahun=1985;

  printf("NAMA          : %s\n",info_rekan.nama);
  printf("Tanggal lahir : %d-%d-
%d",info_rekan.tgl_lahir.tanggal,info_rekan.tgl_lahir.bulan,info_rek
an.tgl_lahir.tahun);
}
Contoh program 2
#include"stdio.h"
#include"conio.h"
#include"string.h"
#include"math.h"

struct{
char nama[20];
char alamat[20];
float gaji;
}pegawai1;
void main()
{char g[15];
float gj;
printf("Nama pegawai 1:");
gets(pegawai1.nama);
printf("Alamat pegawai 1:");
gets(pegawai1.alamat);
printf("Gaji pegawai 1:");
gets(g);
}
```

LATIHAN

1. Struktur dan Fungsi

```
#include"stdio.h"
#include"conio.h"
struct data_tanggal
{int tanggal;
 int bulan;
 int tahun;};

 void cetak_info_tanggal(struct data_tanggal unit_tgl);
void main()
{struct data_tanggal saat_proses = {12,9,1989};

 cetak_info_tanggal(saat_proses);
 getch();
}

void cetak_info_tanggal(struct data_tanggal unit_tgl)
{static char *nama_bulan[]=
 {"Kode bulan salah!",
  "Januari","Februari","Maret","April","Mei",
  "Juni","Juli","Agustus","September","Oktober",
  "November","Desember"};
  printf("%d %s %d\n",unit_tgl.tanggal,
                      nama_bulan[unit_tgl.bulan],
                      unit_tgl.tahun);
}
```

2. Program tukar menggunakan struktur dan fungsi

```
#include <stdio.h>
#include <conio.h>
void tukar_xy(int *x, int *y); /* deklarasi fungsi */

void main()
{ struct koordinat
  { int x; int y; };
  struct koordinat posisi = { 21, 34 };
  clrscr();
  printf("x, y semula --> %d, %d\n", posisi.x, posisi.y);
  tukar_xy(&posisi.x, &posisi.y);
```

```
  printf("x, y kini --> %d, %d\n", posisi.x, posisi.y);
  getch();
}
void tukar_xy(int *x, int *y)
{ int z; z = *x; *x = *y; *y = z; }
   Output setelah program dijalankan
   x, y semula --> 21, 34
   x, y kini --> 34, 21
```

3. Contoh program ADT

Modul berikut adalah untuk manipulasi jam :

```
File Deskripsi isi
jam.h Type dan prototype Jam
jam.c Realisasi (body) dari jam.h
mjam.c Main program untuk mentest beberapa fungsi/prosedur pada jam.c
```

```
/* File : jam.h */
/* deklarasi TYPE dan prototype type jam */
#ifndef jam_H
#define jam_H
typedef struct
{
int HH;
int MM;
int SS;
}
jam;
/* prototype */
void ResetJam (jam * J);
/* Mengisi sebuah jam J dengan 00:00:00 */
void TulisJam (jam J);
/* menulis sebuah jam */
int JamToDetik (jam J);
/* konversi jam ke detik */
jam DetikToJam (int d);
/* konversi dari detik menjadi jam */
#endif
```

```
/* File : jam.c */
/* Body prototype type jam */
#include "jam.h"
/* BODY prototype */
void
ResetJam (jam * J)
/* Mengisi sebuah jam J dengan 00:00:00 */
{ /* KAMUS LOKAL */
/* ALGORITMA */
(*J).HH = 0;
(*J).MM = 0;
(*J).SS = 0;
}
void TulisJam (jam J)
/* menulis sebuah jam */
{ /* KAMUS LOKAL */
/* ALGORITMA */
printf ("Jam : %2d:%2d:%2d\n", J.HH, J.MM, J.SS);
}
int
JamToDetik (jam J)
/* konversi jam ke detik */
{ /* KAMUS LOKAL */
/* ALGORITMA */
return (J.HH * 3600 + J.MM * 60 + J.SS);
}
jam
DetikToJam (int d)
/* konversi dari detik ke struktur jam */
{ /* KAMUS LOKAL */
jam J;
int sisa;
/* ALGORITMA */
J.HH = d / 3600;
sisa = d % 3600;
J.MM = sisa / 60;
J.SS = sisa % 60;
return J;
}
```

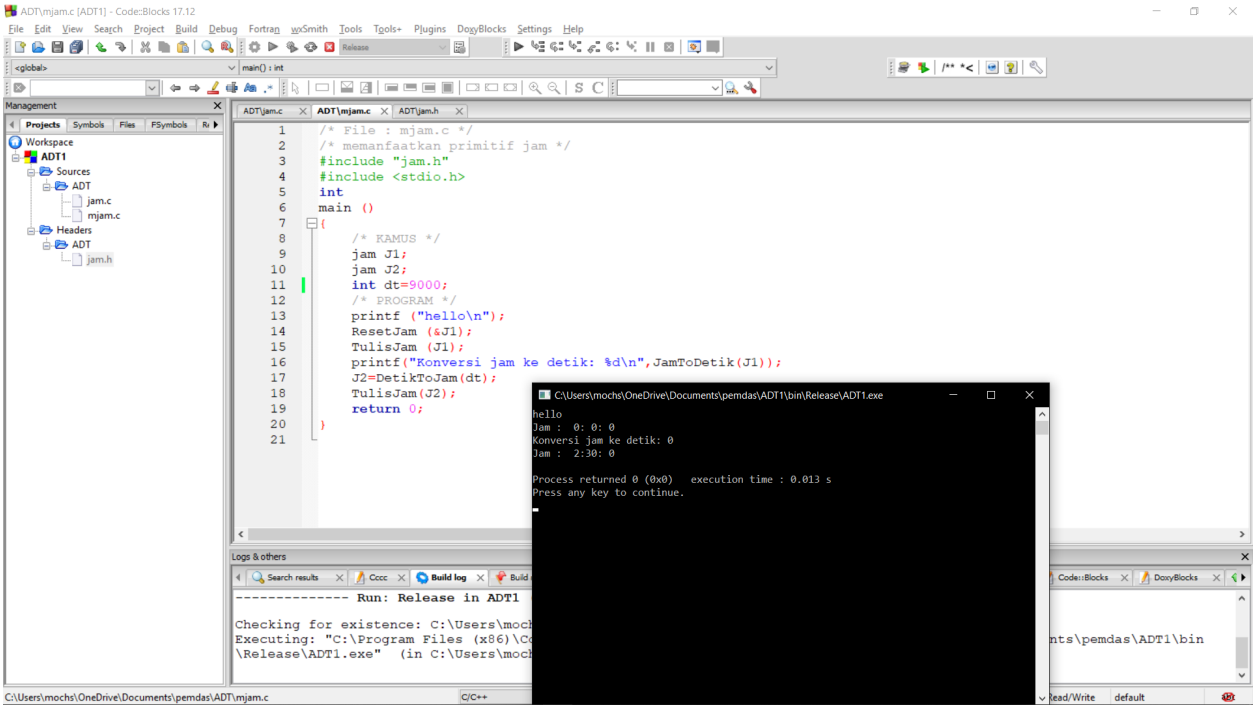
```
/* File : mjam.c */
```

```
/* memanfaatkan primitif jam */
#include "jam.h"
int
main ()
{/* KAMUS */
jam J1;
jam J2;
int dt=1000;
/* PROGRAM */
printf ("hello\n");
ResetJam (&J1);
TulisJam (J1);
printf("Konversi jam ke detik: %d\n",JamToDetik(J1));
J2=DetikToJam(dt);
TulisJam(J2);
return 0;
}
```

Modul berikut adalah untuk manipulasi jam dengan representasi type yang berbda. Perhatikanlah kode jam1.c dan bandingkanlah dengan jam.c. Program utama untuk mentest tidak diberikan dalam catatan ini.

```
File Deskripsi isi
jam1.h Type dan prototype Jam (definisi type berbeda dengan
jam.h)
jam1.c Realisasi (body) dari jam1.h

/* File : jam1.c */
/* Body proto type jam dengan representasi am/pm */
#include "jam1.h"
/* BODY prototype */
void
ResetJam (jam * J)
/* Mengisi sebuah jam J dengan 00:00:00 */
{ /* KAMUS LOKAL */
/* ALGORITMA */
(*J).HH = 0;
(*J).MM = 0;
(*J).SS = 0;
(*J).pasi = pm;
}
void TulisJam (jam J)
/* menulis sebuah jam */
{ /* KAMUS LOKAL */
```

```
/* ALGORITMA */
if (J.pasi == pm)
{
printf ("Jam : %2d:%2d:%2d PM\n", J.HH, J.MM, J.SS);
}
else
{
printf ("Jam : %2d:%2d:%2d AM\n", J.HH, J.MM, J.SS);
};
}
int JamToDetik (jam J)
/* konversi jam ke detik */
{ /* KAMUS LOKAL */
/* ALGORITMA */
if (J.pasi == pm)
{
return (J.HH * 3600 + J.MM * 60 + J.SS);
}
else
{
return ((J.HH + 12) * 3600 + J.MM * 60 + J.SS);
};
}
jam DetikToJam (int d)
/* konversi dari detik ke struktur jam */
{ /* KAMUS LOKAL */
jam J;
int sisa;
/* ALGORITMA */
J.HH = d / 3600;
if (J.HH > 12)
{
J.HH = J.HH - 12;
J.pasi = pm;
}
else
{
J.pasi = am;
};
sisa = d % 3600;
J.MM = sisa / 60;
J.SS = sisa % 60;
return J;
}
```



TUGAS

Buatlah program untuk menginputkan 10 data nilai nama mahasiswa, QUIS, UTS dan UAS dalam suatu struktur daftar nilai mahasiswa dalam suatu kelas mata kuliah pemrograman dasar dan mengoutputkan rata-rata nilainya. Gunakan ADT dan tipe data bentukan.

6. ALGORITMA PENGURUTAN (SORTING

SUB-CPMK 6 : *Menyusun algoritma dan program untuk menyelesaikan masalah saintifik menggunakan pengurutan dan pencarian*

DASAR TEORI

PENGURUTAN

Algoritma Penyortiran digunakan untuk mengatur ulang array atau elemen daftar yang diberikan sesuai dengan operator perbandingan pada elemen. Operator perbandingan digunakan untuk menentukan urutan elemen baru dalam struktur data masing-masing

Buble Sort :

Merupakan algoritma pengurutan paling tua dengan metode pengurutan paling sederhana. Pengurutan yang dilakukan dengan membandingkan masing-masing item dalam suatu list secara berpasangan, menukar item jika diperlukan, dan mengulaginya sampai akhir list secara berurutan, sehingga tidak ada lagi item yang dapat ditukar.

Selection Sort :

Ide utama dari algoritma selection sort adalah memilih elemen dengan nilai paling rendah dan menukar elemen yang terpilih dengan elemen ke-i. Nilai dari i dimulai dari 1 ke n, dimana n adalah jumlah total elemen dikurangi 1.

Insertion Sort :

Algoritma insertion sort pada dasarnya memilah data yang akan diurutkan menjadi dua bagian, yang belum diurutkan dan yang sudah diurutkan. Elemen pertama diambil dari bagian array yang belum diurutkan dan kemudian diletakkan sesuai posisinya pada bagian lain dari array yang telah diurutkan. Langkah ini dilakukan secara berulang hingga tidak ada lagi elemen yang tersisa pada bagian array yang belum diurutkan.

LATIHAN

1. Contoh program untuk bubble sort

```c
/* Bubble sort code */
#include <stdio.h>
int main()
{
  int array[100], n, c, d, swap;

  printf("Enter number of elements\n");
```

```
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  for (c = 0 ; c < n - 1; c++)
  {
    for (d = 0 ; d < n - c - 1; d++)
    {
      if (array[d] > array[d+1]) /* For decreasing order use < */
      {
        swap       = array[d];
        array[d]   = array[d+1];
        array[d+1] = swap;
      }
    }
  }
  printf("Sorted list in ascending order:\n");

  for (c = 0; c < n; c++)
     printf("%d\n", array[c]);
  return 0;
}
```

2. Contoh program untuk selection sort

```
#include <stdio.h>

int main()
{
  int array[100], n, c, d, position, swap;

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  for (c = 0; c < (n - 1); c++)
  {
    position = c;

    for (d = c + 1; d < n; d++)
    {
```

```
      if (array[position] > array[d])
        position = d;
    }
    if (position != c)
    {
      swap = array[c];
      array[c] = array[position];
      array[position] = swap;
    }
  }

  printf("Sorted list in ascending order:\n");

  for (c = 0; c < n; c++)
    printf("%d\n", array[c]);

  return 0;
}
```

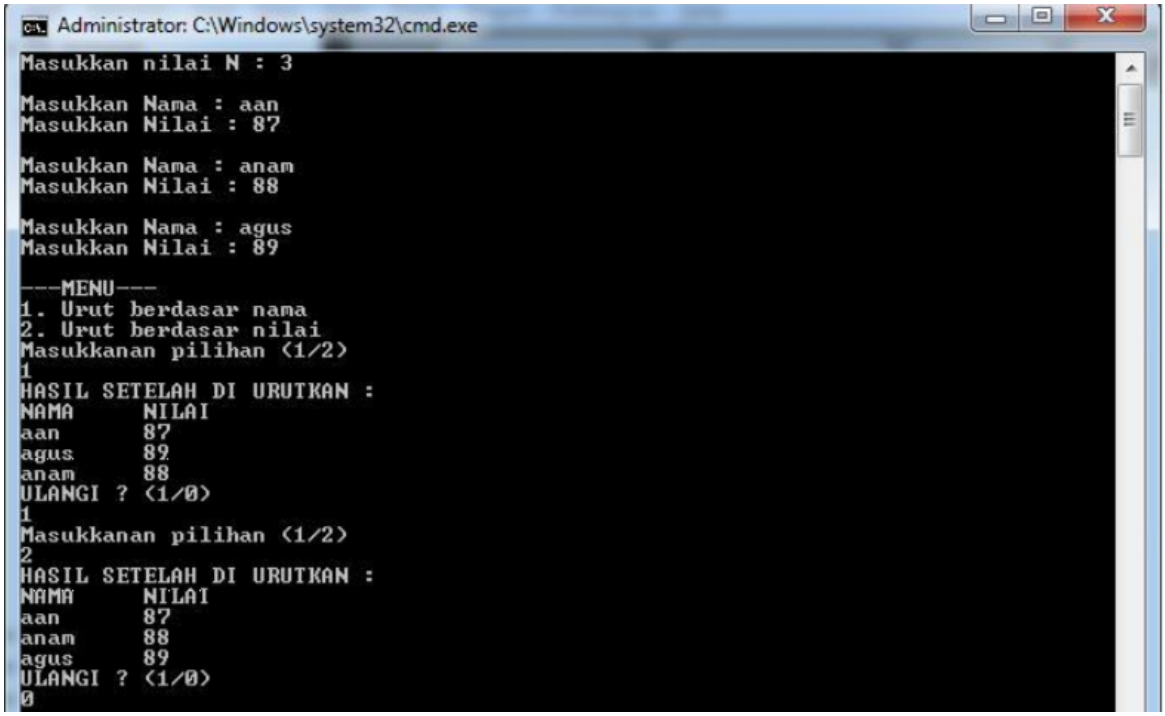
3. Contoh program untuk insertion sort

```
/* Insertion sort ascending order */

#include <stdio.h>

int main()
{
  int n, array[1000], c, d, t;

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  for (c = 1 ; c <= n - 1; c++) {
    d = c;

    while ( d > 0 && array[d-1] > array[d]) {
      t          = array[d];
      array[d]   = array[d-1];
      array[d-1] = t;

      d--;
    }
  }

  printf("Sorted list in ascending order:\n");
```

```
  for (c = 0; c <= n - 1; c++) {
    printf("%d\n", array[c]);
  }

  return 0;
}
```

TUGAS

1. Buatlah sebuah program sorting dengan input dari user, kemudian berikan menu ingin dilakukan sorting menggunakan algoritma apa (Buble, Insertion, Selection).
2. Buatlah program yang dapat melakukan sorting hanya pada range tertentu, beri masukan x dan y, kemudian sorting data hanya pada data ke x sampai data ke y saja. Bebas menggunakan algoritma apapun.
3. Buatlah program yang dapat mengurutkan data nilai mahasiswa di kelasmu, beri input jumlah data kemudian input data nama dan nilai mahasiswa. Tampilkan menu ingin urutkan berdasarkan nama atau nilai, apabila user memilih nama, maka urutkan data tersebut berdasarkan nama apabila memilih nilai maka urutkan berdasarkan nilai kemudian tampilkan data setelah di urutkan, jangan lupa untuk menampilkan data sebelum di urutkan terlebih dahulu



7. ALGORITMA PENCARIAN(SEARCHING)

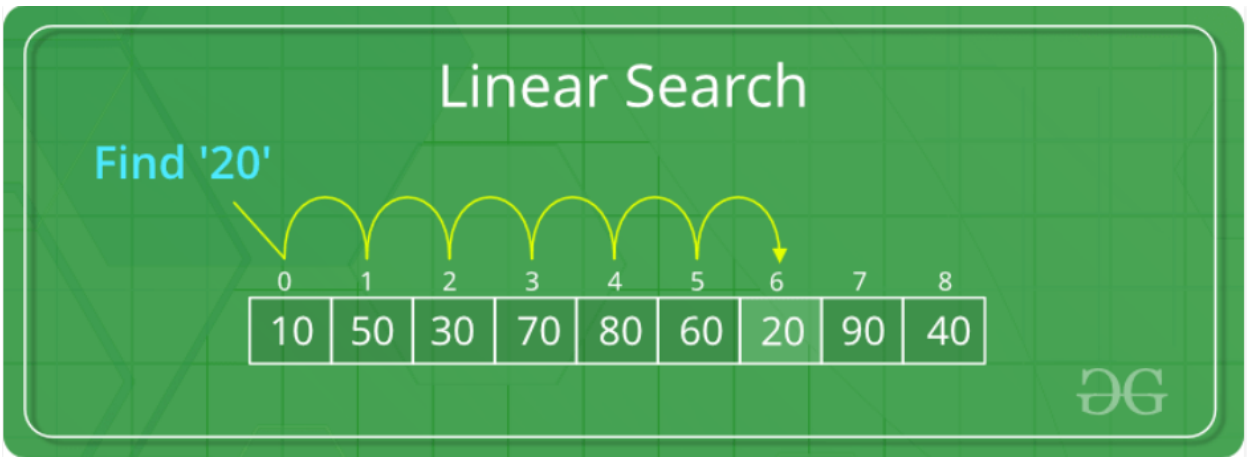
SUB-CPMK 6 : *Menyusun algoritma dan program untuk menyelesaikan masalah saintifik menggunakan pengurutan dan pencarian*

DASAR TEORI

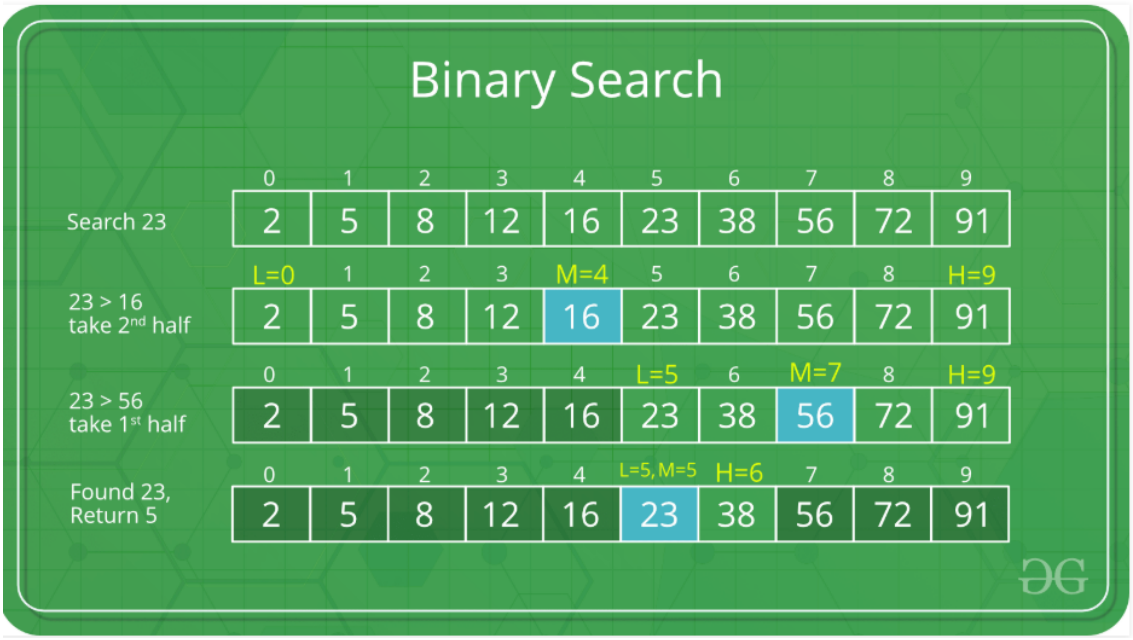
PENCARIAN

Algoritma Pencarian dirancang untuk memeriksa elemen atau mengambil elemen dari struktur data di mana elemen itu disimpan. Berdasarkan pada jenis operasi pencarian, algoritma ini umumnya diklasifikasikan ke dalam dua kategori:

Pencarian Berurutan: Dalam hal ini, daftar atau array dilintasi secara berurutan dan setiap elemen diperiksa. Misalnya: Pencarian Linier untuk menemukan elemen "20" dalam daftar angka yang diberikan :



Pencarian Interval: Algoritma ini secara khusus dirancang untuk pencarian dalam struktur data yang diurutkan. Jenis algoritma pencarian ini jauh lebih efisien daripada Pencarian Linear karena mereka berulang kali menargetkan pusat struktur pencarian dan membagi ruang pencarian menjadi dua. Sebagai Contoh: Pencarian Biner untuk menemukan elemen "23" dalam daftar angka yang diberikan :



LATIHAN

1. Linear search

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.   int array[100], search, c, n;
6.
7.   printf("Enter number of elements in array\n");
8.   scanf("%d", &n);
9.
10.       printf("Enter %d integer(s)\n", n);
11.
12.       for (c = 0; c < n; c++)
13.         scanf("%d", &array[c]);
14.
15.       printf("Enter a number to search\n");
16.       scanf("%d", &search);
17.
18.       for (c = 0; c < n; c++)
19.       {
20.         if (array[c] == search)    /* If required element is
    found */
21.         {
22.           printf("%d is present at location
    %d.\n", search, c+1);
23.           break;
24.         }
25.       }
26.       if (c == n)
```

```
27.          printf("%d isn't present in the array.\n", search);
28.
29.        return 0;
30.      }
```

2. Liner search untuk beberapa penemuan

   Dalam kode di bawah ini kita akan mencetak semua lokasi di mana elemen yang diperlukan ditemukan dan juga berapa kali itu terjadi dalam daftar.

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.  int array[100], search, c, n, count = 0;
6.
7.  printf("Enter number of elements in array\n");
8.  scanf("%d", &n);
9.
10. printf("Enter %d numbers\n", n);
11.
12. for (c = 0; c < n; c++)
13. scanf("%d", &array[c]);
14.
15. printf("Enter a number to search\n");
16. scanf("%d", &search);
17.
18. for (c = 0; c < n; c++) {
19. if (array[c] == search) {
20. printf("%d is present at location %d.\n", search, c+1);
21. count++;
22. }
23. }
24. if (count == 0)
25. printf("%d isn't present in the array.\n", search);
26. else
27. printf("%d is present %d times in the
array.\n", search, count);
28.
29. return 0;
30. }
```

3. Linear search dengan fungsi

```
1. #include <stdio.h>
2.
```

```
3. long linear_search(long [], long, long);
4.
5. int main()
6. {
7.    long array[100], search, c, n, position;
8.
9.    printf("Input number of elements in array\n");
10.      scanf("%ld", &n);
11.
12.      printf("Input %d numbers\n", n);
13.
14.      for (c = 0; c < n; c++)
15.         scanf("%ld", &array[c]);
16.
17.      printf("Input a number to search\n");
18.      scanf("%ld", &search);
19.
20.      position = linear_search(array, n, search);
21.
22.      if (position == -1)
23.         printf("%d isn't present in the array.\n", search);
24.      else
25.         printf("%d is present at location
   %d.\n", search, position+1);
26.
27.      return 0;
28.   }
29.
30.   long linear_search(long a[], long n, long find) {
31.      long c;
32.
33.      for (c = 0 ;c < n ; c++ ) {
34.         if (a[c] == find)
35.            return c;
36.      }
37.
38.      return -1;
39.   }
```

4. Linear search dengan pointer

```
1. long linear_search(long *pointer, long n, long find)
2. {
3.    long c;
4.
5.    for (c = 0; c < n; c++) {
6.       if (*(pointer+c) == find)
7.          return c;
```

```
8.    }
9.
10.      return -1;
11.   }
```

5. Cari kemunculan karakter apa pun di string – strpbrk

```
#include <stdio.h>
#include <string.h>

int main()
{
      char str[] = "teacher teach tea";
      char search[] = "ac";
      char *ptr = strstr(str, search);

      if (ptr != NULL) /* Substring found */
      {
            printf("'%s' contains '%s'\n", str, search);
      }
      else /* Substring not found */
      {
            printf("'%s' doesn't contain '%s'\n", str, search);
      }

      return 0;
}
```

6. Cari string di dalam string lain – strstr

```
#include <stdio.h>
#include <string.h>

int main()
{
      char str[] = "finding digits where there could be digit 5236 is
amazing";
      char digits[] = "0123456789";
      char *ptr = strpbrk(str, digits);

      if (ptr != NULL) /* Expected character is found */
      {
            printf("'%s' contains at least one character from '%s'\n",
str, digits);
      }
      else /* Expected character isn't found */
      {
```

```
            printf("'%s' doesn't contain any character from '%s'\n",
str, digits);
      }

      return 0;
}
```

TUGAS

1. Buatlah sebuah program Diberikan sebuah masukkan Array *S* berupa kata sebanyak *N*. Anda diminta untuk mencetak semua isi Array *S*yang diakhiri dengan kata "api", misal untuk kata "sapi" maka akan ditampilkan karena bagian belakangnya mengandung kata "api".

   Input Format

   - Baris pertama adalah nilai *N* yang merupakan banyaknya isi Array *S*
   - Baris kedua adalah isi array *S* yang dipisahkan oleh enter dan seterusnya sebanyak *N*.

   Constraints

   $1 \leq N \leq 100$

   Output Format

   Isi array *S* yang diakhiri kata 'api'

   Sample Input 0

   ```
   5
   gunung
   merapi
   kereta
   sapi
   betina
   ```

   Sample Output 0

   ```
   merapi
   sapi
   ```

2. Buatlah program dengan syarat dibawah :
   - Diberikan sebuah masukkan Array *B* yang berisi elemen tipe bentukan buku sebanyak *N* dan sebuah masukkan *cariNama* yang merupakan nama pengarang.

   - Tipe bentukan buku berisi pengarang, judul, dan tahun.

- Anda diminta untuk mencetak semua isi Array *B* yang memiliki nama pengarang sama dengan nama masukkan *cariNama* dan tahun di atas 2005.

Input Format

- Baris pertama adalah nilai *N* yang merupakan banyaknya isi Array *B*
- Baris kedua adalah isi array *B* yang setiap elemen tipe bentukkannya dipisahkan oleh enter dan seterusnya sebanyak *N*.
- Baris terakhir adalah masukkan *cariNama*

Constraints

$1 \leq N \leq 100$

Output Format

Isi array *B* secara berurutan: pengarang, judul, dan tahun yang dipisahkan oleh **koma**, dengan ketentuan nama pengarang sesuai dengan masukkan *cariNama* dan tahun di atas 2005.

Sample Input 0

```
3
Anto
pengenalan oop
2009
Antonio
algoritma pemrograman
2006
indra
bisnis inteligen
2007
Anto
```

Sample Output 0

```
Anto,pengenalan oop,2009
Antonio,algoritma pemrograman,2006
```

Explanation 0

- Baris terakhir input adalah masukkan *cariNama* .
- Output adalah data buku yang nama pengarangnya sesuai dengan *cariNama* yang diinputkan pada baris terakhir, yaitu "supri" dan tahunnya di atas tahun 2005.

- Output pengarang, judul dan tahun dicetak dengan dipisahkan oleh tanda **koma** .

3. Rubah program pada no.2 menjadi program dengan file , gunakan menu!.